Codici Segreti

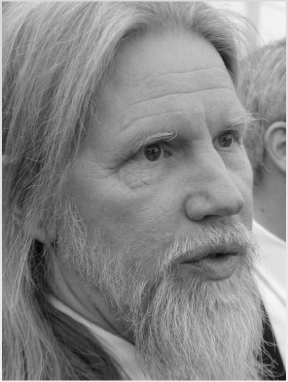
Agostino Dovier

Dip di Scienze Matematiche, Informatiche, e Fisiche Univ. di Udine

10 GENNAIO 2018

Crittografia a chiave pubblica

Diffie e Hellman (Turing Award 2015)



- Nel 1976 Whitfield Diffie e Martin Hellman pubblicano una metodologia rivoluzionaria per la comunicazione cifrata.
- Alla stessa idea erano giunti prima Malcolm J. Williamson, James H. Ellis, Clifford C. Cocks nei primi anni 70. Lavorando al *Government Communications Head Quarter (GB)*, il loro lavoro fu secretato (e all'epoca giudicato tecnologicamente impraticabile)
- In cosa consiste?

Crittografia a chiave pubblica

Diffie e Hellman

- Nel 1976 Whitfield Diffie e Martin Hellman pubblicano una metodologia rivoluzionaria per la comunicazione cifrata.
- Alla stessa idea erano giunti prima Malcolm J. Williamson, James H. Ellis, Clifford C. Cocks nei primi anni 70. Lavorando al *Government Communications Head Quarter (GB)*, il loro lavoro fu secretato (e all'epoca giudicato tecnologicamente impraticabile)
- In cosa consiste?

- Nel 1976 Whitfield Diffie e Martin Hellman pubblicano una metodologia rivoluzionaria per la comunicazione cifrata.
- Alla stessa idea erano giunti prima Malcolm J. Williamson, James H. Ellis, Clifford C. Cocks nei primi anni 70. Lavorando al *Government Communications Head Quarter (GB)*, il loro lavoro fu secretato (e all'epoca giudicato tecnologicamente impraticabile)
- In cosa consiste?

La crittografia a chiave pubblica poggia le sue fondamenta sulla comunicazione privata tra due personaggi chiave.

Crittografia a chiave pubblica

Idee generali

Bob ed Alice

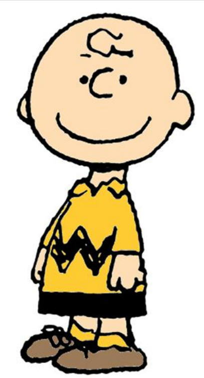
Crittografia a chiave pubblica

Idee generali

Bob ed Alice

Inoltre c'è bisogno della spia.

Crittografia a chiave pubblica
Idee generali



Charlie

Crittografia a chiave pubblica
Idee generali

Charlie's

- Alice e Bob (e tutti) pubblicano (in un elenco telefonico, negli anni ’70, in rete oggi) una volta per sempre la loro chiave pubblica (K_A quella di Alice, K_B quella di Bob, etc.), nota a tutti.
- Alice vuole inviare il messaggio m a Bob.
- Alice codifica il messaggio m per Bob con la chiave pubblica di Bob (K_B) e invia il messaggio cifrato $COD(m, K_B)$.
- Bob riceve il messaggio $COD(m, K_B)$ e usa la sua chiave privata H_B per decodificare il messaggio.

- Le chiavi pubblica e privata sono progettate in modo tale che
$$DEC(COD(m, K_B), H_B) = m$$
- L'operazione di decifrazione, sapendo la chiave privata dev'essere algoritmicamente facile
- L'operazione di decrittazione (per Charlie) deve essere algoritmicamente impraticabile.
- Anche se l'impresa è possibile: conoscendo K_B e $c = COD(m, K_B)$ si possono generare uno ad uno i messaggi di lunghezza opportuna, $m_1, m_2, \ldots, m_\ell$.
- Dunque si codificano uno ad uno con la chiave K_B e si vede se $COD(m_i, K_B) = c$.

Algoritmo di Euclide

Calcoliamo il massimo comun divisore MCD

Assumiamo $a > b$ in chiamata (se $a = b$ allora $\text{MCD}(a, b) = a$; se $a < b$ li scambio):

$$
\begin{array}{l}
\text{MCD}(a, b) \\
\quad \text{if} \;\; b = 0 \\
\quad \text{return} \;\; a \\
\quad \text{else} \;\; \text{return} \;\; \text{MCD}(b, a \underline{\bmod} b)
\end{array}
$$

Algoritmo di Euclide

MCD(a, b)
    if $b = 0$
    return a
    else return MCD$(b, a \underline{\text{mod}}\ b)$

Correttezza: (1) Se $d|a$ e $d|b$ allora $a = kd$ e $b = hd$ per qualche coppia di interi h, k. Dobbiamo mostrare che $d|b$ (già nelle ipotesi) e che $d|a \underline{\text{mod}}\ b$. Ora, sappiamo che $a = bc + a \underline{\text{mod}}\ b$ per qualche c, per definizione della divisione con resto. Pertanto

$$kd = hdc + a \underline{\text{mod}}\ b \Rightarrow a \underline{\text{mod}}\ b = d(k - hc)$$

dunque $d|a \underline{\text{mod}}\ b$

Algoritmo di Euclide

MCD(a, b)
if $b = 0$
return a
else return MCD($b, a \,\underline{\mathrm{mod}}\, b$)

Completezza: (2) Supponiamo ora $d|b$ e $d|a \,\underline{\mathrm{mod}}\, b$, dunque che $b = kd$ e $a \,\underline{\mathrm{mod}}\, b = hd$ per h, k opportuni. Poichè $a = bc + a \,\underline{\mathrm{mod}}\, b$, per qualche c, vale che

$$a = kdc + hd = d(kc + h)$$

e dunque $d|a$ ($d|b$ era un'ipotesi).

Algoritmo di Euclide

$$
\begin{array}{l}
\mathrm{MCD}(a,b) \\
\quad \text{if } \; b = 0 \qquad \text{return } \; a \\
\quad \text{else } \; \text{return } \; \mathrm{MCD}(b, a \,\underline{\text{mod}}\, b)
\end{array}
$$

Complessità: Sia consideri la successione di Fibonacci: $F_0 = 1, F_1 = 1, F_{k+2} = F_{k+1} + F_k$.
Mostriamo (induz. su k): Se MCD(a,b) esegue almeno $k \geq 1$ chiamate ricorsive, allora $a \geq F_{k+1}$ e $b \geq F_k$.
Con $k = 1$ significa che $a > b > 0$. Ovvero $b \geq 1 = F_1$ e $a \geq 2 = F_2$.
Supponiamo ora che vengano eseguite $k+1$ chiamate.
Ciò significa che MCD$(b, a \,\underline{\text{mod}}\, b)$ necessita di k chiamate. Per ipotesi induttiva, $b \geq F_{k+1}$ e $a \,\underline{\text{mod}}\, b \geq F_k$.
Rimane da mostrare che $a \geq F_{k+2}$. Sappiamo che $a > b \geq F_k$. Ma $a = bh + a \,\underline{\text{mod}}\, b$ per qualche h.
Essendo che $a > b$, $h \geq 1$. Dunque $a \geq b + a \,\underline{\text{mod}}\, b \geq F_{k+1} + F_k = F_{k+2}$.

Algoritmo di Euclide

MCD(a, b)
  if $b = 0$
  return a
  else return MCD($b, a \underline{\bmod} b$)

Da questo risultato segue il Teorema di Lamè che dice che, se $a > b \geq 0$ e $b < F_k$, allora l'esecuzione di MCD(a, b) necessita di meno di k chiamate ricorsive. Poichè F_k cresce esponenzialmente

$$\left(\lim_{n \to \infty} \frac{F_n}{F_{n-1}} = \frac{1 + \sqrt{5}}{2} \right)$$

il numero di chiamate (e di conseguenza la complessità dell'algoritmo di Euclide) è $O(\log b)$.

Algoritmo di Euclide Esteso

L'algoritmo di Euclide può essere esteso per trovare i coefficienti interi x ed y tali che

$$ax + by = \mathrm{MCD}(a, b)$$

L'algoritmo seguente restituisce la tripla $\langle d, x, y\rangle$ ove $d = \mathrm{MCD}(a, b)$.

$$\begin{array}{l}
\mathrm{EE}(a, b)\\
\quad \text{if } b = 0\\
\quad \text{return } \langle a, 1, 0\rangle\\
\quad \text{else begin}\\
\qquad \langle d', x', y'\rangle \leftarrow \mathrm{EE}(b, a \underline{\mathrm{mod}}\, b);\\
\qquad \text{return } \langle d', y', x' - (a \underline{\mathrm{div}}\, b)y'\rangle\\
\quad \text{end}
\end{array}$$

Ad esempio: $\mathrm{EE}(30, 24) = \langle 6, 1, -1\rangle$, $\mathrm{EE}(144, 64) = \langle 16, 1, -2\rangle$.

Algoritmo di Euclide Esteso

Correttezza. Sappiamo già che d è il massimo comun divisore tra a e b. Per induzione sul numero di chiamate ricorsive mostriamo che x e y si comportano nel modo atteso. Con 0 chiamate ricorsive significa che $b = 0$. Ma allora abbiamo che

$$a1 + b0 = a = \mathrm{MCD}(a, b)$$

Prendiamo il caso induttivo. Sappiamo che $d' = \mathrm{MCD}(a, a \,\underline{\mathrm{mod}}\, b) = \mathrm{MCD}(a, b)$. Sappiamo per ipotesi induttiva che $bx' + (a \,\underline{\mathrm{mod}}\, b)y' = d'$. Dobbiamo mostrare che $ay' + b(x' - (a \,\underline{\mathrm{div}}\, b)y') = d'$. Ricordiamoci che $a = (a \,\underline{\mathrm{div}}\, b)b + (a \,\underline{\mathrm{mod}}\, b)$. Allora

$$\begin{array}{lcl} ay' + b(x' - (a \,\underline{\mathrm{div}}\, b)y') & = & \\ (a \,\underline{\mathrm{div}}\, b)by' + (a \,\underline{\mathrm{mod}}\, b)y' + bx' - b(a \,\underline{\mathrm{div}}\, b)y' & = & \\ (a \,\underline{\mathrm{mod}}\, b)y' + bx' & = & d' \end{array}$$

Algoritmo di Euclide Esteso

Complessità. E' evidente che EE(a, b) eredita totalmente i risultati di complessità di MCD(a, b) (facendo uno studio preciso, non asintotico, le costanti ovviamente cambiano leggermente).

A cosa può mai servire questo algoritmo? Supponiamo a sia un numero primo, $a > b$ (dunque, in particolare, MCD$(a, b) = 1$). Eseguendo EE(a, b) ottengo (in tempo $O(\log b)$) una tripla $\langle 1, x, y \rangle$ tale che $ax + by = 1$. Se ragiono nel campo finito $\mathbb{Z}_a$, y è l'inverso (rispetto a '$\cdot$') di b. Dunque *EE* mi permette di trovare in modo molto efficiente gli inversi in $\mathbb{Z}_a$.

Algoritmo di Euclide Esteso

Complessità. E' evidente che EE(a, b) eredita totalmente i risultati di complessità di MCD(a, b) (facendo uno studio preciso, non asintotico, le costanti ovviamente cambiano leggermente).

A cosa può mai servire questo algoritmo? Supponiamo a sia un numero primo, $a > b$ (dunque, in particolare, MCD$(a, b) = 1$). Eseguendo EE(a, b) ottengo (in tempo $O(\log b)$) una tripla $\langle 1, x, y \rangle$ tale che $ax + by = 1$. Se ragiono nel campo finito $\mathbb{Z}_a$, y è l'inverso (rispetto a '$\cdot$') di b. Dunque *EE* mi permette di trovare in modo molto efficiente gli inversi in $\mathbb{Z}_a$.

- Si considerino n interi positivi $(a_1, \ldots, a_n)$
- E un intero c.
- Il problema è quello di trovare $i_1 < i_2 < \cdots < i_k$ con $i_j \in \{1, \ldots, n\}$ tali che (o di dire che non esistono)

$$a_{i_1} + \cdots + a_{i_k} = c \qquad (1)$$

- L'idea del fusto (o zaino, da Trapdoor Knapsack ma un fusto cilindrico rende meglio l'idea) è quella di riuscire ad impilare gli oggetti a_i giusti l'uno sull'altro in modo da raggiungere l'altezza esatta c.

- In generale si tratta di trovare un vettore di n Booleani $\vec{m}$ tale che

$$\sum_{i=1}^{n} a_i m_i = c \qquad (2)$$

- L'idea di Merkle e Hellman è di codificare un messaggio di n bits $\vec{m} = b_1 \cdots b_n$ con il numero c.
- Stabilire l'esistenza di $\vec{m}$ che soddisfa il vincolo (2) è NP-completo.

- Una n-upla $(a_1, \ldots, a_n)$ è supercrescente se
$$(\forall j \in \{2, \ldots, n\}) \left(\sum_{i=1}^{j-1} a_i < a_j \right)$$
- Il problema del fusto con n-uple supercrescenti è lineare.
- Esempio: La n-upla è $(2, 3, 6, 12, 24, 48)$. Sia $c = 33$. 48 non puo' essere, essendo maggiore. 24 ci deve essere in quanto con tutti i precedenti si puo' fare solo 23. Rimangono 9. Dunque 12 non puo' essere. Ma 6 ci deve essere. Rimangono 3. 3 ci deve essere. Rimane 0.
- Il messaggio $\vec{m} = (0, 1, 1, 0, 1, 0)$.

- Si sceglie un intero n, una n-upla supercrescente $\vec{a} = (a_1, \ldots, a_n)$, e un intero $u > 2a_n$.
- Si sceglie un intero $v < u$ e primo con u $(MCD(u, v) = 1)$
- Si calcola v^{-1} nell'aritmetica in modulo u
- Esiste perchè $MCD(u, v) = 1$, si calcola facilmente con Euclide Esteso (EE in breve).
- Si calcola la n-upla $\vec{d} = (d_1, \ldots, d_n)$ dove

$$d_i = va_i \mod u$$

- Possiamo dire $\vec{d} = v\vec{a} \mod u$.
- Anche se $\vec{a}$ è supercrescente, $\vec{d}$ è una lista qualunque.
- Il numero n ed il vettore $\vec{d}$ sono resi pubblici.

Crittografia a chiave pubblica

Il cifrario del fusto — Generazione chiavi

- Si sceglie un intero n, una n-upla supercrescente $\vec{a} = (a_1, \ldots, a_n)$, e un intero $u > 2a_n$.
- Si sceglie un intero $v < u$ e primo con u $(MCD(u, v) = 1)$
- Si calcola v^{-1} nell'aritmetica in modulo u
- Esiste perchè $MCD(u, v) = 1$, si calcola facilmente con Euclide Esteso (EE in breve).
- Si calcola la n-upla $\vec{d} = (d_1, \ldots, d_n)$ dove

$$d_i = va_i \mod u$$

- Possiamo dire $\vec{d} = v\vec{a} \mod u$.
- Anche se $\vec{a}$ è supercrescente, $\vec{d}$ è una lista qualunque.
- Il numero n ed il vettore $\vec{d}$ sono resi pubblici.

- Alice vuol spedire un messaggio (Booleano) a Bob.
- Vede la chiave pubblica $n_B, \vec{d}_B$ di Bob.
- Codifica il suo messaggio a blocchi di n_B bits. Consideriamo un blocco (chiamiamolo $\vec{m}$)
- Alice calcola

$$c = \vec{m}\vec{d}_B = \sum_{i=1}^{n} m_i d_i$$

- Dunque lo spedisce a Bob

Crittografia a chiave pubblica

Il cifrario del fusto — Decifrazione

- Bob ha ricevuto $c = \vec{m}\vec{d}_B$
- Effettua dunque i seguenti calcoli (facili per lui)

$$
\begin{aligned}
c^* &= v^{-1}c \mod u \\
&= v^{-1}(\vec{d}_B\vec{m}) \mod u \\
&= v^{-1}(v\vec{a}\vec{m}) \mod u \\
&= (v^{-1}v)(\vec{a}\vec{m}) \mod u \\
&= 1(\vec{a}\vec{m}) \mod u \\
&= \vec{a}\vec{m} \mod u
\end{aligned}
$$

- Si osservi che $\vec{a}$ supercrescente, $u > 2a_n$ per costruzione, dunque $\vec{a}\vec{m} < u$. Possiamo togliere il modulo.
- A Bob non resta altro che risolvere il problema del fusto $\vec{a}\vec{m} = c^*$, con la successione supercrescente $\vec{a}$ (tempo lineare).

- Charlie ha ricevuto $c = \vec{m}\vec{d}_B$
- Charlie conosce n e $\vec{d}_B$ (sa che è stato inviato a Bob).
- Deve calcolare $\vec{m}$.
- Per lui è un problema di Knapsack e dunque NP-completo.
- Forzato da Shamir nel 1982. Ma non dimostra che P=NP (non risolve tutti i Knapsack in P ma solo quelli generati in quel modo: lavora su ritrasformare $\vec{d}$ in $\vec{a}$).

- Charlie ha ricevuto $c = \vec{m}\vec{d}_B$
- Charlie conosce n e $\vec{d}_B$ (sa che è stato inviato a Bob).
- Deve calcolare $\vec{m}$.
- Per lui è un problema di Knapsack e dunque NP-completo.
- Forzato da Shamir nel 1982. Ma non dimostra che P=NP (non risolve tutti i Knapsack in P ma solo quelli generati in quel modo: lavora su ritrasformare $\vec{d}$ in $\vec{a}$).

Qualche promemoria

- Per $n \in \mathbb{N} \setminus \{0\}$ definiamo $\equiv_n$ (congruo modulo n):

$$a \equiv_n b \text{ se e solo se } (a \underline{\bmod} n) = (b \underline{\bmod} n)$$

- Se $ab \equiv_n ac$ e $\text{MCD}(a, n) = 1$, allora $b \equiv_n c$
- Corollario, se $ab \equiv_n 0$ e $\text{MCD}(a, n) = 1$, allora $b \equiv_n 0$.
- Il *Piccolo teorema di Fermat*: se p è primo e $p \not| a$, allora $a^{p-1} \equiv_p 1$.
- La *funzione di Eulero* Φ viene definita come segue:

$$\Phi(n) = |\{z \in \mathbb{N} : 0 < z < n, \text{MCD}(n, z) = 1\}|$$

- Il *Teorema di Eulero*: se $\text{MCD}(a, n) = 1$, allora $a^{\Phi(n)} \equiv_n 1$.

L'ESPONENZIALE FINITO (DISCRETO)

Vogliamo calcolare $2^{8438} \underline{\text{mod}}\ 789$. Con $\log_2(8438)$ divisioni calcoliamo la rappresentazione binaria di $8438=8192+128+64+32+16+4+2$.
Calcoliamo le potenze di due fino al valore desiderato:

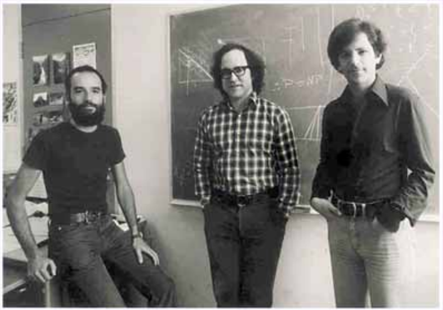
i	$2^i \underline{\text{mod}}\ 789$
1	2
2	$2^2 = 4$
4	$4^2 = 16$
8	$16^2 = 256$
16	$256^2 = 65536 \equiv_{789} 49$
32	$49^2 = 2401 \equiv_{789} 34$
64	$34^2 = 1156 \equiv_{789} 367$
128	$367^2 = 134689 \equiv_{789} 559$
256	$559^2 = 312481 \equiv_{789} 37$
512	$37^2 = 1369 \equiv_{789} 580$
1024	$580^2 = 336400 \equiv_{789} 286$
2048	$286^2 = 81796 \equiv_{789} 529$
4096	$529^2 = 279841 \equiv_{789} 535$
8192	$535^2 = 286225 \equiv_{789} 607$

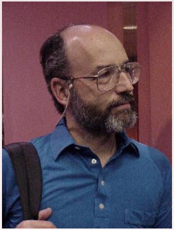
Ad ogni passo facciamo il quadrato del numero ottenuto al passo precedente e lo modularizziamo, mantenendo sempre numeri "piccoli" (operazioni in tempo costante, fissato il 789). A questo punto si tratta di fare al più $\log_2(8438)$ moltiplicazioni in modulo:

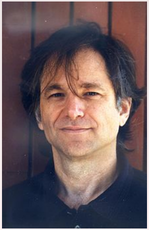
$$\begin{aligned}
2^{8438} &= (((((2^{8192}2^{128} \underline{\text{mod}}\ 789)2^{64} \underline{\text{mod}}\ 789)2^{32} \underline{\text{mod}}\ 789)2^{16} \underline{\text{mod}}\ 789)2^{4} \underline{\text{mod}}\ 789)2^{2} \underline{\text{mod}}\ 789 \\
&= (((((607 \cdot 559 \underline{\text{mod}}\ 789)367 \underline{\text{mod}}\ 789)34 \underline{\text{mod}}\ 789)49 \underline{\text{mod}}\ 789)16 \underline{\text{mod}}\ 789)4 \underline{\text{mod}}\ 789 \\
&= 109
\end{aligned}$$

Crittografia a chiave pubblica

Rivest, Shamir, e Adleman (Turing award 2002)







1. Bob sceglie due numeri primi p e q.
   - Per generare un primo:
   - Si prende un numero dispari delle dimensioni desiderate
   - Si vede se è primo
     Manindra Agrawal, Neeraj Kayal, Nitin Saxena, PRIMES is in P. Annals of Mathematics 160(2):781–793, 2004 (RR 2002)
   - Nota bene fin d'ora. Una cosa è stabilire se è primo, un'altra è trovare i suoi fattori se non lo è.
   - Se lo è OK, altrimenti lo si incrementa di 2 e si ripete.
   - I primi sono infiniti (e densi $\frac{\lg n}{n}$). Dopo un po' lo si trova.
2. Bob calcola $n = pq$
3. Bob calcola $\Phi(n) = (p-1)(q-1)$

1. Bob sceglie due numeri primi p e q.
   - Per generare un primo:
   - Si prende un numero dispari delle dimensioni desiderate
   - Si vede se è primo
     Manindra Agrawal, Neeraj Kayal, Nitin Saxena, PRIMES is in P. Annals of Mathematics 160(2):781–793, 2004 (RR 2002)
   - Nota bene fin d'ora. Una cosa è stabilire se è primo, un'altra è trovare i suoi fattori se non lo è.
   - Se lo è OK, altrimenti lo si incrementa di 2 e si ripete.
   - I primi sono infiniti (e densi $\frac{\lg n}{n}$). Dopo un po' lo si trova.
2. Bob calcola $n = pq$
3. Bob calcola $\Phi(n) = (p-1)(q-1)$

Crittografia a chiave pubblica

RSA – Generazione chiavi

1. Bob sceglie due numeri primi p e q.
   - Per generare un primo:
   - Si prende un numero dispari delle dimensioni desiderate
   - Si vede se è primo
     Manindra Agrawal, Neeraj Kayal, Nitin Saxena, PRIMES is in P. Annals of Mathematics 160(2):781–793, 2004 (RR 2002)
   - Nota bene fin d'ora. Una cosa è stabilire se è primo, un'altra è trovare i suoi fattori se non lo è.
   - Se lo è OK, altrimenti lo si incrementa di 2 e si ripete.
   - I primi sono infiniti (e densi $\frac{\lg n}{n}$). Dopo un po' lo si trova.
2. Bob calcola $n = pq$
3. Bob calcola $\Phi(n) = (p-1)(q-1)$

Crittografia a chiave pubblica
RSA – Generazione chiavi

1. Bob sceglie due numeri primi p e q.
   - Per generare un primo:
   - Si prende un numero dispari delle dimensioni desiderate
   - Si vede se è primo
     Manindra Agrawal, Neeraj Kayal, Nitin Saxena, PRIMES is in P. Annals of Mathematics 160(2):781–793, 2004 (RR 2002)
   - Nota bene fin d'ora. Una cosa è stabilire se è primo, un'altra è trovare i suoi fattori se non lo è.
   - Se lo è OK, altrimenti lo si incrementa di 2 e si ripete.
   - I primi sono infiniti (e densi $\frac{\lg n}{n}$). Dopo un po' lo si trova.
2. Bob calcola $n = pq$
3. Bob calcola $\Phi(n) = (p-1)(q-1)$

4. Bob sceglie un altro numero (esponente) e tale che

$$MCD(e, \Phi(n)) = 1$$

   - Sceglie un numero dispari delle dimensioni opportune
   - Esegue l'algoritmo di Euclide tra e e $\Phi(n)$
   - Se l'output è 1, OK,
   - Altrimenti incrementa di due e ritenta
   - Il processo termina al più al numero primo successivo che non divide né $p-1$ né $q-1$.
5. Bob calcola d tale che $de = 1 \mod \Phi(n)$. (si può fare direttamente al passo precedente usando EE in luogo di Euclide)
6. Bob pubblica la chiave (n, e).

1. Alice vuole inviare un messaggio a Bob, conoscendo la chiave pubblica di Bob (n, e).
2. Spezza il messaggio in blocchi (stringhe binarie) tali che la loro interpretazione come numero sia quella di un numero $m < n$ (basta fissare blocchi di lunghezza $\lfloor \lg_2 n \rfloor$).
3. Considera dunque un blocco alla volta.

4. Alice calcola $c = m^e \mod n$
   - L'esponenziale finito è una operazione semplice algoritmicamente.
   - Sia $k = \lfloor lg_2 e \rfloor + 1$.
   - $m = m^{(2^0)}$. Calcolo

$$m^{(2^1)}, m^{(2^2)}, m^{(2^3)}, m^{(2^4)}, \dots, m^{(2^k)}$$

     tutti modulo n. Ciò garantisce che i numeri siano tutti $< e$ (e solo temporaneamente tra e e e^2).
   - Scrivo e in base 2 $(e_k, e_{k-1}, \cdots, e_1, e_0)$, e moltiplico tra loro (sempre in modulo e, sempre con numeri "controllati") i vari $m^{(2^i)}$ tali che $e_i = 1$
5. Alice spedisce c a Bob.

Crittografia a chiave pubblica

RSA – Decifrazione

1. Bob riceve $c = m^e \mod n$.
2. Bob conosce d t.c. $de = 1 \mod \Phi(n)$
3. Bob calcola

$$c^d \mod n = (m^e)^d \mod n = m^{ed} \mod n$$

4. Ci possono essere due casi.
   4.1 $MCD(m, n) = 1$ (essendo n prodotto di due primi, questo fatto è molto probabile).
   4.2 $MCD(m, n) \neq 1$ (meno probabile, ma va considerato).

Sia $MCD(m,n) = 1$

- Per il Teorema di Eulero

$$m^{\Phi(n)} = 1 \mod n \tag{3}$$

- Per costruzione, da EE so trovare h tale che:

$$de + h\Phi(n) = 1 \tag{4}$$

Sia $k = -h$.

- Dunque

$$\begin{array}{rcll} c^d & = & m^{ed} \mod n & \\ & = & m^{k\Phi(n)+1} \mod n & (4) \\ & = & m\,(m^{\Phi(n)})^k \mod n & \\ & = & m\,1^k \mod n & (3) \\ & = & m & \text{Poiché } m < n \end{array}$$

Sia ora $MCD(m,n) \neq 1$

- n è prodotto di due primi p e q; $m < n$ per costruzione.
- Lemma. Se $p \neq q$ sono due primi, $a = b \mod p$ e $a = b \mod q$, allora $a = b \mod pq$.
  Dim. Mostriamo prima che se $a = 0 \mod p$ e $a = 0 \mod q$ allora $a = 0 \mod pq$.
  $a = 0 \mod p \to \exists h\,(a = hp)$
  $a = 0 \mod q \to \exists k\,(a = kq)$
  Poiché p e q primi ed entrambi fattori di a, allora $\exists u(h = uq)$. Dunque $a = upq$ e $a = 0 \mod pq$.
  Ora, se $a = b \mod p$ e $a = b \mod q$ allora $a - b = 0 \mod p$ e $a - b = 0 \mod q$, pertanto $a - b = 0 \mod pq$ dunque $a = b \mod pq$.

Codici Segreti
A. Dovier
Crittografia a chiave pubblica
Euclide
Trapdoor Knapsack
Introduzione
Generazione chiavi
Cifrazione
Decifrazione
Decrittazione
Esponenziale Discreto
RSA
Introduzione
Generazione chiavi
Cifrazione
Decifrazione
Decrittazione
PGP
ElGamal
Conclusioni

Sia ora $MCD(m, n) \neq 1$

- n è prodotto di due primi p e q; $m < n$ per costruzione.
- Lemma. Se $p \neq q$ sono due primi, $a = b \mod p$ e $a = b \mod q$, allora $a = b \mod pq$.
- $MCD(m, n) \neq 1$ significa che $(p|m)$ oppure $(q|m)$ (ma non entrambi, anche se non è importante)
- Sappiamo per costruzione che $de = 1 \mod \Phi(n)$ dunque $\Phi(n)|de - 1$.
- Ma $\Phi(n) = (p - 1)(q - 1) = \Phi(p)\Phi(q)$
- Dunque $\Phi(p)\Phi(q)|de - 1$, pertanto vale sia $\Phi(p)|de - 1$ che $\Phi(q)|de - 1$, ovvero $de = h\Phi(p) + 1$ e $de = k\Phi(q) + 1$ per h e k opportuni.

- Se $MCD(m,p)=1$, per Teorema di Eulero $m^{\Phi(p)}=1 \mod p$, dunque $m^{de}=m(m^{\Phi(p)})^h=m \mod p$
- Se $MCD(m,p)\neq 1$, siamo nel caso $(p|m)$. Dunque $m=pr$ per r opportuno. Pertanto

$$\begin{array}{rcl} m^{de} & = & (pr)^{de} \\ & = & p^{de}r^{de} \\ & = & 0 \mod p \\ & = & pr \mod p \\ & = & m \mod p \end{array}$$

- Similmente, sia con $MCD(m,q)=1$ che $MCD(m,q)=1$, vale che $m^{de}=m \mod q$
- Dunque, per il Lemma sopra, in ogni caso $m^{de}=m \mod pq$

1. Charlie intercetta $c = m^e \mod n$.
2. Charlie sa che Alice l'ha inviato a Bob e conosce e e n pubblicati da Bob.
3. Al solito, in principio, può generare tutti i messaggi $m_1, m_2, \ldots, m_t$ di lunghezza opportuna (sono in numero finito, dato n), effettuare la codifica e vedere se trova c (forza bruta). È impraticabile.
4. La strada più semplice (in apparenza) è scoprire i fattori p e q di n. Ma anche questo non lo sappiamo fare (per ora) con algoritmi polinomiali in $\log n$.
5. Charlie, per ora, non decritta il messaggio.

1. Charlie intercetta $c = m^e \mod n$.
2. Charlie sa che Alice l'ha inviato a Bob e conosce e e n pubblicati da Bob.
3. Al solito, in principio, può generare tutti i messaggi $m_1, m_2, \ldots, m_t$ di lunghezza opportuna (sono in numero finito, dato n), effettuare la codifica e vedere se trova c (forza bruta). È impraticabile.
4. La strada più semplice (in apparenza) è scoprire i fattori p e q di n. Ma anche questo non lo sappiamo fare (per ora) con algoritmi polinomiali in $\log n$.
5. Charlie, per ora, non decritta il messaggio.

1. Charlie intercetta $c = m^e \mod n$.
2. Charlie sa che Alice l'ha inviato a Bob e conosce e e n pubblicati da Bob.
3. Al solito, in principio, può generare tutti i messaggi $m_1, m_2, \ldots, m_t$ di lunghezza opportuna (sono in numero finito, dato n), effettuare la codifica e vedere se trova c (forza bruta). È impraticabile.
4. La strada più semplice (in apparenza) è scoprire i fattori p e q di n. Ma anche questo non lo sappiamo fare (per ora) con algoritmi polinomiali in $\log n$.
5. Charlie, per ora, non decritta il messaggio.

1. Charlie intercetta $c = m^e \mod n$.
2. Charlie sa che Alice l'ha inviato a Bob e conosce e e n pubblicati da Bob.
3. Al solito, in principio, può generare tutti i messaggi $m_1, m_2, \ldots, m_t$ di lunghezza opportuna (sono in numero finito, dato n), effettuare la codifica e vedere se trova c (forza bruta). È impraticabile.
4. La strada più semplice (in apparenza) è scoprire i fattori p e q di n. Ma anche questo non lo sappiamo fare (per ora) con algoritmi polinomiali in $\log n$.
5. Charlie, per ora, non decritta il messaggio.

PROBLEMA:

TROVARE UN DIVISORE (NON UNITARIO) DI UN NUMERO n

fattorizza(n)

for $i = 2$ to $\sqrt{n}$

if n *mod* $i = 0$

print(i è un divisore di n)

exit

Nel caso peggiore compie un numero di passi pari a $\sqrt{n}$.

Possiamo togliere i pari, i multipli di 3, i multipli di 5, ma la sostanza non cambia.

Se n è un numero di 100 cifre, ogni divisione mi costa $10^{-10}s$, e dispongo di 10^9 processori, mi serve tempo

$$\approx \frac{10^{-10}\sqrt{10^{100}}}{3600 \cdot 24 \cdot 365 \cdot 10^9} = 3 \cdot 10^{23} \text{ANNI}$$

PROBLEMA:

TROVARE UN DIVISORE (NON UNITARIO) DI UN NUMERO n

fattorizza(n)
  for $i = 2$ to $\sqrt{n}$
    if $n \ mod \ i = 0$
      print(i è un divisore di n)
      $exit$

Nel caso peggiore compie un numero di passi pari a $\sqrt{n}$.
Possiamo togliere i pari, i multipli di 3, i multipli di 5, ma la sostanza non cambia.
Se n è un numero di 100 cifre, ogni divisione mi costa $10^{-10}s$, e dispongo di 10^9 processori, mi serve tempo

$$\approx \frac{10^{-10}\sqrt{10^{100}}}{3600 \cdot 24 \cdot 365 \cdot 10^9} = 3 \cdot 10^{23} \text{ANNI}$$

PROBLEMA:
TROVARE UN DIVISORE (NON UNITARIO) DI UN NUMERO n

fattorizza(n)
  for $i = 2$ to $\sqrt{n}$
    if n mod $i = 0$
      print(i è un divisore di n)
      $exit$

Nel caso peggiore compie un numero di passi pari a $\sqrt{n}$.
Possiamo togliere i pari, i multipli di 3, i multipli di 5, ma la sostanza non cambia.
Se n è un numero di 100 cifre, ogni divisione mi costa $10^{-10}s$, e dispongo di 10^9 processori, mi serve tempo

$$\approx \frac{10^{-10}\sqrt{10^{100}}}{3600 \cdot 24 \cdot 365 \cdot 10^9} = 3 \cdot 10^{23} \text{ANNI}$$

Problema:

trovare un divisore (non unitario) di un numero n

fattorizza(n)
    for $i = 2$ to $\sqrt{n}$
        if $n\ mod\ i = 0$
            print(i è un divisore di n)
            $exit$

Nel caso peggiore compie un numero di passi pari a $\sqrt{n}$.
Possiamo togliere i pari, i multipli di 3, i multipli di 5, ma la sostanza non cambia.
Se n è un numero di 100 cifre, ogni divisione mi costa $10^{-10}s$, e dispongo di 10^9 processori, mi serve tempo

$$\approx \frac{10^{-10}\sqrt{10^{100}}}{3600 \cdot 24 \cdot 365 \cdot 10^9} = 3 \cdot 10^{23} \text{ANNI}$$

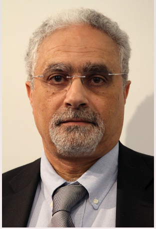
PGP E SCAMBIO DELLE CHIAVI

- Nel 1991Phil R. Zimmerman sviluppa PGP (Pretty Good Privacy)
- Cifrari a chiave privata (AES, DES) e Chiave Pubblica (RSA, ElGamal) vengono usati in combinazione.
- Usando RSA ci si passa la chiave (corta) per l'AES, e si usa AES (più veloce) per cifrare il messaggio (lungo).

Zimmeman ebbe una grande idea, senza secondi fini. Fu sottoposto a diversi processi negli USA.

ELGAMAL E IL LOGARITMO DISCRETO

- Nel 1985 Taher Elgamal architetta un altro sistema di cifratura a chiave pubblica.
- La *forza* del sistema di cifratura è basata sulla difficoltà computazionale di calcolare il logaritmo in un campo finito (in breve *logaritmo discreto*).



ELGAMAL E IL LOGARITMO DISCRETO

- Sia p numero primo *grande* (100 cifre o più), sia α radice primitiva in $\mathbb{Z}_p$ (ovvero $\alpha^{p-1} = 1$ e $\alpha^j \neq 1$ per $j < p - 1$).
- Bob sceglie a caso da $X_B \in \mathbb{Z}_p \setminus \{0\}$ e genera e rende nota la propria chiave pubblica $Y_B = \alpha^{X_B} \mod p$
- Alice vuole inviare m a Bob (ove $m < p$ — se il messaggio è lungo lo spezziamo).
- Alice sceglie a caso $k \in \mathbb{Z}_p \setminus \{0\}$. Calcola:
  1. $K_A = (Y_B)^k \mod p$
  2. $C_1 = \alpha^k \mod p$
  3. $C_2 = m \cdot K_A \mod p$
- Comunica dunque $\langle C_1, C_2 \rangle$ a Bob

ELGAMAL E IL LOGARITMO DISCRETO

- Sia p numero primo *grande* (100 cifre o più), sia α radice primitiva in $\mathbb{Z}_p$ (ovvero $\alpha^{p-1} = 1$ e $\alpha^j \neq 1$ per $j < p - 1$).
- Bob sceglie a caso da $X_B \in \mathbb{Z}_p \setminus \{0\}$ e genera e rende nota la propria chiave pubblica $Y_B = \alpha^{X_B} \mod p$
- Alice vuole inviare m a Bob (ove $m < p$ — se il messaggio è lungo lo spezziamo).
- Alice sceglie a caso $k \in \mathbb{Z}_p \setminus \{0\}$. Calcola:
  1. $K_A = (Y_B)^k \mod p$
  2. $C_1 = \alpha^k \mod p$
  3. $C_2 = m \cdot K_A \mod p$
- Comunica dunque $\langle C_1, C_2 \rangle$ a Bob

ELGAMAL E IL LOGARITMO DISCRETO

- Bob riceve $\langle C_1, C_2 \rangle$
- Bob calcola $K_B = (C_1)^{X_B} \mod p$ e dunque $\hat{m} = (C_2 \cdot K_B^{-1}) \mod p$.
- Funziona?
- Notate che (tutto $\mod p$): $K_B = (C_1)^{X_B} = (\alpha^k)^{X_B} = (\alpha^{k \cdot X_B}) = (\alpha^{X_B})^k = Y_B^k = K_A$.
- E dunque (tutto $\mod p$): $\hat{m} = (C_2 \cdot K_B^{-1}) = (m \cdot K_A \cdot K_B^{-1}) = (m \cdot K_A \cdot K_A^{-1}) = m$.

Attacchi a ElGamal

(alla lavagna)

- Birthday Attack
- Baby Step / Giant Step (BSGS)
- Algoritmo di Pohlig-Hellman
- Index calculus
- Gli attacchi sopra funzionano in certe situazioni. Per avere sicurezza p deve essere molto grande e $p - 1$ deve avere (anche) dei divisori primi grandi.
- I codici ellittici migliorano la situazione garantendo la stessa sicurezza con dimensioni minori.

Attacchi a ElGamal

(alla lavagna)

- Birthday Attack
- Baby Step / Giant Step (BSGS)
- Algoritmo di Pohlig-Hellman
- Index calculus
- Gli attacchi sopra funzionano in certe situazioni. Per avere sicurezza p deve essere molto grande e $p-1$ deve avere (anche) dei divisori primi grandi.
- I codici ellittici migliorano la situazione garantendo la stessa sicurezza con dimensioni minori.

Crittografia a chiave pubblica

Scambio delle chiavi — Diffie e Hellman 1976

- Alice e Bob vogliono concordare una chiave di un DES o di un AES.
- Alice e Bob scelgono assieme e pubblicano un numero primo p ed un numero $\alpha \in \{2, \dots, p-1\}$
- Alice sceglie $A \in \mathbb{N}$ e calcola e comunica $\alpha^A \mod p$
- Bob sceglie $B \in \mathbb{N}$ e calcola e comunica $\alpha^B \mod p$
- La chiave da usarsi è $\alpha^{AB} \mod p$
- Alice sa calcolare $\alpha^{AB} \mod p = (\alpha^B)^A \mod p$
- Bob sa calcolare $\alpha^{AB} \mod p = (\alpha^A)^B \mod p$
- Charlie, che intercetta tutto, conosce $\alpha, p, \alpha^A, \alpha^B$, ma NON sa calcolare α^{AB}.
- Di nuovo, un modo per farlo è calcolare il logaritmo discreto.

Crittografia a chiave pubblica

Scambio delle chiavi — Diffie e Hellman 1976

- Alice e Bob vogliono concordare una chiave di un DES o di un AES.
- Alice e Bob scelgono assieme e pubblicano un numero primo p ed un numero $\alpha \in \{2, \ldots, p-1\}$
- Alice sceglie $A \in \mathbb{N}$ e calcola e comunica $\alpha^A \mod p$
- Bob sceglie $B \in \mathbb{N}$ e calcola e comunica $\alpha^B \mod p$
- La chiave da usarsi è $\alpha^{AB} \mod p$
- Alice sa calcolare $\alpha^{AB} \mod p = (\alpha^B)^A \mod p$
- Bob sa calcolare $\alpha^{AB} \mod p = (\alpha^A)^B \mod p$
- Charlie, che intercetta tutto, conosce $\alpha, p, \alpha^A, \alpha^B$, ma NON sa calcolare α^{AB}.
- Di nuovo, un modo per farlo è calcolare il logaritmo discreto.

- Alice e Bob vogliono concordare una chiave di un DES o di un AES.
- Alice e Bob scelgono assieme e pubblicano un numero primo p ed un numero $\alpha \in \{2, \ldots, p-1\}$
- Alice sceglie $A \in \mathbb{N}$ e calcola e comunica $\alpha^A \mod p$
- Bob sceglie $B \in \mathbb{N}$ e calcola e comunica $\alpha^B \mod p$
- La chiave da usarsi è $\alpha^{AB} \mod p$
- Alice sa calcolare $\alpha^{AB} \mod p = (\alpha^B)^A \mod p$
- Bob sa calcolare $\alpha^{AB} \mod p = (\alpha^A)^B \mod p$
- Charlie, che intercetta tutto, conosce $\alpha, p, \alpha^A, \alpha^B$, ma NON sa calcolare α^{AB}.
- Di nuovo, un modo per farlo è calcolare il logaritmo discreto.

Crittografia a chiave pubblica

Scambio delle chiavi — Diffie e Hellman 1976

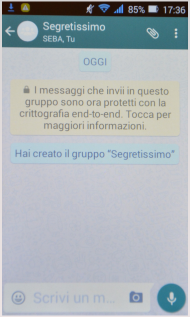
- Alice e Bob vogliono concordare una chiave di un DES o di un AES.
- Alice e Bob scelgono assieme e pubblicano un numero primo p ed un numero $\alpha \in \{2, \ldots, p-1\}$
- Alice sceglie $A \in \mathbb{N}$ e calcola e comunica $\alpha^A \mod p$
- Bob sceglie $B \in \mathbb{N}$ e calcola e comunica $\alpha^B \mod p$
- La chiave da usarsi è $\alpha^{AB} \mod p$
- Alice sa calcolare $\alpha^{AB} \mod p = (\alpha^B)^A \mod p$
- Bob sa calcolare $\alpha^{AB} \mod p = (\alpha^A)^B \mod p$
- Charlie, che intercetta tutto, conosce $\alpha, p, \alpha^A, \alpha^B$, ma NON sa calcolare α^{AB}.
- Di nuovo, un modo per farlo è calcolare il logaritmo discreto.

CONCLUSIONI

- Ovviamente la nostra visita della crittografia è ben lungi dall'essere esaustiva
- Se avete passione, approfondire un po' sarà facile (magari date un'occhiata ai vari MD4, MD5, WPA, WEP, etc).

Conclusioni

La sicurezza . . . quotidiana



Siamo immersi in un mondo connesso e con molti malintenzionati.
La crittografia a chiave pubblica ci permette di comunicare in sicurezza.
Sempre che non si sappia fattorizzare n in p e q o calcolare $\log_\alpha \beta$ in $\mathbb{Z}_p$.